UNIX & LINUX

SPONSORED BY aws

Capture all the output of a script to a file (from the script itself) [duplicate]

Asked 4 years, 5 months ago Modified 2 years, 1 month ago Viewed 71k times

17

1

This question already has answers here:

Redirect all subsequent commands' stderr using exec (4 answers)

Closed 4 years ago.

I have a bash script that calls various commands and prints some output (both from the called commands themselves, such as `git pull`, and informative messages generated by the script itself such as `Operation took XX minutes`.

I'd like to capture the whole output to a file **from the script itself**: basically I'm trying to avoid the need to call `./myscript.sh | tee file.txt` for *non-relevant-here* reasons.

Basically I'd like to do something like this:

```
startCapture

git pull

echo "Text"

other-command

endCapture
```

I also require the output to be printed on my shell while the script is running.

The final goal is to:

1. run `./myscript.sh` without additional shell constructs
2. see the output on the terminal as I do now
3. obtain a file on disk with the whole output

Is this even possible?

bash files string output

Share Improve this question Follow

edited Jun 1, 2020 at 9:34

Dan Dascalescu
6,660 4 17 24

asked Feb 16, 2018 at 17:06

Dr. Gianluigi Zane Zanettini
412 1 4 9

1 For those that end up here from Google, you might also consider this answer (unix.stackexchange.com/a/164017/103505) to just put all the code you want to output in a function in the script and then redirect the output of that function. – cjbarth Apr 15, 2020 at 13:22

3 Answers

Sorted by: Highest score (default)

5

You can always invoke `script` inside a script to log everything.

As to print and log everything at the same time in a bash script to `log.txt`:

```
#!/bin/bash

if [ -z "$SCRIPT" ]
then
    /usr/bin/script log.txt /bin/bash -c "$0 $*"
    exit 0
fi

echo teste
```

Seeing the log `log.txt`:

```
$ ./a.sh
Script started, output file is log.txt
teste

Script done, output file is log.txt
$ cat log.txt
Script started on Fri Feb 16 17:57:26 2018
command: /bin/bash -c ./a.sh
teste

Script done on Fri Feb 16 17:57:26 2018
```

Share Improve this answer Follow

edited Feb 16, 2018 at 18:05

answered Feb 16, 2018 at 17:19
user avata Rui F Ribeiro
53.5k 25 138 216

7 If I understand correctly (some explanations would help :), the script is trying to detect if it's running under `script`, and not re-execute itself. As of Ubuntu 20, it looks like `$SCRIPT` is not set, so the script goes into an infinite loop. – Dan Dascalescu Jun 1, 2020 at 10:06

3 Good idea to use `script` though, because it allows for interactive terminal output, e.g. Carriage Return, to work. In effect, commands that display single-line progress output (archivers for example), behave normally. As for solving the infinite loop problem, we can test the shell level: `if (( $SHLVL < 3 )); then script...; fi` . – Dan Dascalescu Jun 1, 2020 at 10:35

Also, the arguments of the scripts end up being concatenated and passed in the *code* argument to the shell started by `script` (not `/bin/bash`, that `/bin/bash` argument is ignored with util-linux `script`) – Stéphane Chazelas Jul 24, 2020 at 8:52

2

A method I found to capture all output from any session is to start a new bash session and tee to a log file. its really useful for tracking more then just a script.

```
bash | tee ~/bash.log #this will save standard output until the bash session is ended
bash | tee ~/bash.log  2>&1 #this will save all output including errors until the bash session is ended
```

or you can just tee the script it's self

```
./myscript.sh | tee ./myscript.log #this will log only the output of the script.
```

Share Improve this answer Follow

edited Feb 16, 2018 at 17:30

answered Feb 16, 2018 at 17:10

thebtm
1,219 2 10 18

Your solution addresses STDOUT, but not STDERR – user1404316 Feb 16, 2018 at 17:28

I put in an example with `2>&1` – thebtm Feb 16, 2018 at 17:31

I don't think this answers the question because requirement #1 was "run ./myscript.sh **without additional shell constructs**". – Dan Dascalescu Jun 1, 2020 at 9:36

This question was edited 6 hours ago, I didn't have that information when I first answered the question. It was using `>` output instead of tee and that is why I shared the option for tee. – thebtm Jun 1, 2020 at 16:04

2

You want to use tee.

Ex:

```
echo "Hello World" | tee out.txt
```

This creates a file out.txt with the output from the command and prints it to the screen. Use "tee -a filename" if you want to append to the file.

```
echo "Hello" | tee -a out.txt
echo "World" | tee -a out.txt
```

out.txt will have two lines Hello and World (without -a it would only have world)

If you want to save the entire script and output the entire script:

```
./script.sh | tee output.txt
```

Share Improve this answer Follow

answered Feb 16, 2018 at 17:15
Mark
39 2

1 Your solution addresses STDOUT, but not STDERR – user1404316 Feb 16, 2018 at 17:26

unix.stackexchange.com/a/61932/268823 explains how to do STDOUT and STDERR – Mark Feb 17, 2018 at 21:17

I don't think this answers the question because requirement #1 was "run ./myscript.sh **without additional shell constructs**". – Dan Dascalescu Jun 1, 2020 at 9:36

I guess one way to do it would be to put everything into a function, and use tee at the end. That would effectively work as a start and stop capture. And redirecting the STDERR if that is the intention. – Mark Jul 6, 2020 at 17:01